Pocket Computer $1000 (June 1977)

5th Generation Computer Announced (July 1977)

49

Annual Forecast for the Coming Decade in Computing

Pending demise of PL/1 becomes evident (June 1977)

Cost per line of producing complex software exceeds $50 (Sept 1977)

Number of makers of large main frames drops to 5 (Nov 1977)

Superprogrammers rate 40K salaries (Feb 1978)

The pocket computer is available at $500 (June 1978)

A new largest prime number is found (July 1978)

There are more than 35,000 home-built computers in the U.S. (Aug 1978)

Giant computer embezzlement scheme is revealed, involving government funds (Nov 1978)

Pocket computers with increased capability are now $250 (Dec 1978)

The pocket calculator industry reaches 35 million units per year (March 1979)

Cost of executed instructions falls below 100 million per dollar (June 1979)

Automobile on-board computers begin (Sept 1979)

More than 50% of U.S. computing power is in small dedicated machines (March 1980)

Computer kits disappear, replaced by prepackaged units (April 1980)

Cost of executed instructions falls below one billion per dollar (March 1981)

Number of persons employed full time as "programmers" starts to drop (June 1981)

Superprogrammers rate 50K salaries (Dec 1981)

Computing courses are required in more than half of all U.S. high schools (April 1982)

Personal computers exceeding the power of the 7094 cost $2500 (June 1982)

Restrictive legislation on use of data banks is enacted (Dec 1982)

The job category "programmer" has disappeared (August 1983)

Superprogrammers rate 70K salaries (Jan 1984)

Cost per line of producing complex software exceeds $100 (March 1984)

Cost of executed instructions falls below 10 billion per dollar (March 1985)

Last traditional textbook publisher gives up (June 1985)

Top speeds of super computers reach one nanosecond add time (July 1985)

Transmission of hard copy mail falls to 50% of 1977 level (Feb 1986)

Number of centralized large computers in the U.S. falls below 1000 (June 1986)

Dr. John W. Wrench, Jr., notes that the values for π^N in our N-Series for N = 2, 38, and 45 are not rounded properly in their last digit. The last digit shown in each case should be one larger.

He also notes that the value given on the cover of issue No. 12 (the first 600 digits of the sine of one radian) is in error in its last two digits. They appear as 34 and should read 20.

We regret that any of these four errors may have caused computational troubles for our readers. □

AN OLD PROBLEM REVISITED

One glass contains 100 cc of wine; a second glass contains 100 cc of water. One cc of wine is moved to the second glass, and then one cc of the mixture is moved back to the first glass. How does the amount of wine now in the first glass compare to the amount of water now in the second glass?

The amounts are equal: 99.00990099 cc in each case. If the whole operation is repeated, the amounts will be 98.0394079 cc and, after a third state, 97.0881325 cc. This figure will approach 50 cc after many such transfers of the liquids.

Problem: How many complete transfers will it take to bring the figure down to 51 cc? To 50.5 cc? To 50.05 cc?

PROBLEM 165

POPULAR COMPUTING is published monthly at Box 272, Calabasas, California 91302. Subscription rate in the United States is $20.50 per year, or $17.50 if remittance accompanies the order. For Canada and Mexico, add $1.50 per year to the above rates. For all other countries, add $3.50 per year to the above rates. Back issues $2.50 each. Copyright 1977 by POPULAR COMPUTING.

Editor: Audrey Gruenberger
Publisher: Fred Gruenberger
Associate Editors: David Babcock
Irwin Greenwald

Contributing editors: Richard Andree
William C. McGee
Thomas R. Parkin

Advertising Manager: Ken W. Sims
Art Director: John G. Scott
Business Manager: Ben Moore

Contest 11 Result

Contest 11 (issue No. 42) called for writing a program in ANSI Fortran to do the following: read a card bearing nine 3-digit numbers in columns 1-27. Check that each of these numbers is in the range 100 to 999 and print an error message if any data item is out of range; otherwise, print the nine numbers, their sum, the sum of squares, the mean, and the standard deviation (the latter two correct to 3 decimal places). Nine lines of test data were furnished with the problem.

The winning solution would be the one using the least number of characters in the program, with the characters on each line of code counted up to the last card column used by the compiler.

The winner is Joan Farmer, The Polytechnic of London, who wrote: "How difficult 'simple' things are! And in particular, how difficult it is to understand what's in the mind of setters of simple contests. So with Contest 11. Does the setter really want the test data to print 099 or will space99 do? And does he realize that the number 8910285 [the sum of squares for one of the sets of test data given] causes overflow on a 24-bit machine? An ANSI Fortran 4 program that avoids overflow without knowing the size of the machine word on which it is to be run is hardly to be classified as 'simple.' Anyway, for better or worse I decided to assume the answer No to both questions."

Her program, reproduced below, contains 342 characters.

```
C     PRØGRAM TØ READ SETS ØF 9 NUMBERS AND CØMPUTE THEIR SUM(J),
C     SUM ØF SQUARES (K), MEAN (R), AND STANDARD DEVIATION (S).
C     THE 9 NUMBERS ARE PUNCHED ØN ØNE CARD IN CØLUMNS 1-27 AND
C     MUST LIE IN THE RANGE 100-999.   NØ CHECK IS MADE FØR
C     NUMBERS > 999 SINCE THERE IS NØ WAY ØF DETECTING THEM.
C     THE PRØGRAM TERMINATES BY RUNNING ØUT ØF DATA.

      CØMMØNN(9)
    1 J=0
      K=0
      READ(5,7)N
      DØ3I=1,9
      L=N(I)
      IF(L.GT.99)GØTØ2
      WRITE(6,8)N
      GØTØ1
    2 J=J+L
    3 K=K+L*L
      R=FLØAT(J)/9.
      S=SQRT((FLØAT(K)-R*(FLØAT(2*J)-9.*R))/8.)
      WRITE(6,9)N,J,K,R,S
      GØTØ1
    7 FØRMAT(9I3)
    8 FØRMAT(9I4,14H  ØUT ØF RANGE)
    9 FØRMAT(9I4,I6,I9,2F9.3)
      END
```

More on Assemblers

In the Art of Computing series, essay No. 12 (issue No. 41, page 8) was on Assemblers. In issue No. 42, Associate Editor Irwin Greenwald presented his views on the same subject. The discussion continues.

We should be able to agree on some basic principles:

1. The overriding criterion for an assembler (or any machine/language combination for that matter) should be utility for the user. If the system doesn't help the user far more than it impedes him in his work, it is not doing its basic job.

2. As a corollary, the system should do nothing wrong (as in the first precept of the Hippocratic oath: Do no harm).

[For example, the assembler for the GE-225 would accept the following:

```
CØN    DEC    37Ø5
```

squeeze out the illegal "oh" and double what was left, producing the value 750 in the word addressed at CØN, with no error message.]

3. As far as possible, the system should not let the user do anything wrong; that is, it should attempt to guard against user stupidity.

4. The system should facilitate the production of readable, maintainable, comprehensible code.

Granted that, in today's world, it is difficult to isolate the functions that properly belong to the assembler itself from those that are the domain of the link editor, the loader, and the operating system. The user, however, sees all these entities as a cohesive system to use in translating his logic (as expressed perhaps in a flowchart) into running machine language code. If he does everything right, he will have no problems with the system. But he doesn't do everything right (at least most of us don't); it is more common to do many things wrong, and the fun of the game is the tracking down of the errors. The whole point of the discussion is the extent to which the system is helpful in this process.

Let me digress to cite an example taken from Fortran. Except for the necessary JCL, here is a complete Fortran code:

```
1     6|7                                   72
-------+-------------------------------------
       |A = B
       |PRINT A
       |CALL EXIT
```

Most Fortrans will accept that code, compile it, and execute it.* A number will be printed for A--any old value lying around in storage. What should happen is

a) A courteous message that says "You haven't defined B, chum."

b) Another message that says "Execution inhibited."

[Note: no one writes a program with such an obvious blunder. But in working with a long program over a period of months or years, it is easy to assume that B had been defined.]

This is an example of a violation of principle 3. It is much easier to have such things in compilers (and Fortran is full of them) than in assemblers, but the point is still valid. The user wants the system to be on his side in the problem solving process. He also wants, and should get, an unobtrusive system, transparent to him.

Part of Mr. Greenwald's argument is that the writer of the system has to compromise opposing goals in order to make the product acceptable to a broad range of users. The problem is that the compromises are not made evident; the user is not aware of what has been done to him; he is expected to conform to the result.

Consider, for example, the plea for some permissiveness in an assembler, specifically the capability of using, say, any of the mnemonics M, MUA, MUL, MU for the operation MULTIPLY. True, modern assemblers allow the user to specify such synonyms. If that is done, who else can then read his programs?

All too often, it seems, the user is presented with a system and told effectively to live with it; to get used to its idiosyncrasies, to learn to program around its weaknesses, and not to be a chronic complainer. The point of the original article was that too little work has gone into fashioning workable tools that would aid the problem solving process rather than being part of it.

The following is suggested as a checklist of items to consider for a good assembler:

*WATFOR will not.

1. Ease of teaching; ease and speed of learning. Assuming that a new user is familiar with at least one other assembly system, how long will it take him to master <u>this</u> assembler? To what extent will the teacher (which may be a reference manual) be able to point out the conventional assembly features, and to what extent will the teacher have to say "Now watch out for this feature--it's tricky"?

For example, in teaching COMPASS (an assembler for Control Data machines), it is necessary to spell out the rule for forming symbols:

* 8 characters or less
* First character alphabetic
* No embedded punctuation symbols

which would be reasonably simple and straightforward if one didn't then have to add:

* Except that decimal points are permitted within the symbol.

Some bright student will surely ask "Why did they include that exception?" and what does one answer? The true answer is probably that the writer of the assembler found it convenient for <u>his</u> purposes to allow that exception, and every user suffers as a result.

2. Minimize the number of things that have to be looked up to use the system. The use (in COMPASS) of AZJ,LT as a mnemonic for JUMP ON MINUS has already been cited. The corresponding mnemonic for JUMP ON PLUS as AZJ,GE--such things are inexcusable. The user is forced to keep lists of DOs and DON'Ts in front of him, in order to write even simple programs.

Why not try this experiment? Take a dozen competent programmers who all understand the theory of assembly language coding but who have not used this new assembler. Give them some minimal instruction in the new system. Then give them coding sheets and a flowchart for a problem solution (for example, the merging flowchart from PC34-17) and have them code from that flowchart. Record all the questions they ask. Repeat the whole experiment with some other assembler, and compare the results. Even if the differences are difficult to evaluate, a study of the questions asked might indicate the need for significant reworking of the entire package.

3. Check the ease of readability, changeability, and maintainability of programs written with the assembler by experienced people.

4. Check the ability of the system to conform to the basic principles that were listed earlier. To what extent will it be necessary to have one or more system "mothers" in each installation, who are the resident experts on what can go wrong, who can actually decode the garbage that appears on the symbolic listings, and who can help other programmers track down their troubles?

5. To what extent does the system prevent the user from errors? For example, suppose the assembler permits the use of

LOAD ACCUMULATOR (zero level) XXXXX

where the X's indicate a decimal number to be loaded. Since the X number is part of the instruction, it must be restricted to some small value, say that that can be contained in 15 bits, which is less than 32768. If the user attempts to use an X larger than 32768, will the system warn him, as it should?

6. Is there a high correlation between the number of errors in a line of code and the number of error messages? In many assemblers, it is possible to make 4 errors in one instruction, which ought to lead to 4 error messages.

What is needed, before a new system is released to users, is a destruct team, whose function is to make every error possible--to try to circumvent the system--to insure that the system is indeed a useful tool and not merely a necessary evil written to satisfy minimum requirements for the sale of a new machine.

Problem Solution

In issue No. 42, Problem 139 called for finding A and B such that

$$A^B = B^A$$

with the example

$$A = 9/4$$

$$B = 27/8$$

It was intended, but not stated, that A and B should be rational numbers. The following is from Jeff Shallit, Wynnewood, Pennsylvania:

We are looking for solutions of $x^y = y^x$. Let $y = ax$, then

$$x^{ax} = (ax)^x$$

$$x^a = ax$$

$$x^{a-1} = a$$

$$x = a^{1/(a-1)} \qquad (R)$$

$$y = a \cdot a^{1/(a-1)}$$

$$= a^{a/(a-1)} \qquad (S)$$

Thus, picking an appropriate value for a, we may obtain as many solutions as desired. If, given a value for x, we desire the corresponding values for y, we can solve equation (R) by Newton's method to get the corresponding value of a, and then we have $y = ax$.

Or, we can transform equations (R) and (S) as follows.

$$\text{Let } a = 1 + 1/r$$

$$x = (1 + 1/r)^r$$

$$y = (1 + 1/r)^{r+1}$$

For $r = 1$, we get $x = 2$, $y = 4$ (the only non-trivial integral solution). For $r = 2$, we get $x = 9/4$ and $y = 27/8$. For $r = 3$, the values for x and y are 64/27 and 256/81, and so on.

The problem also brought forth comments from Edwin S. Levitan, Hughes Aircraft Company:

If Elapsed Solution Time is defined as the difference between the time at which solution was attained and the time the problem was posed, then I claim a record for Problem 139 of EST = -23 years.

In the summer of 1953 I became intrigued with the same problem. The occasion was a self-study of the solution of differential equations prior to the course I would be taking that Fall. This expression was listed at the rear of my book as being the solution to one of the problem statements.

At first glance, the expression appears to be an abstruse way of describing a straight line at 45° to the axes; indeed, A = B is a solution.

Upon noting that $2^4 = 4^2$, I was then mesmerized into generating additional solution pairs, whose locus looks somewhat like an equilateral hyperbola asymptotic to the lines A = 1 and B = 1. (Note that this was in the pre-calculator days, and I admit to having spent numerous hours of trial and error with a table of logarithms and a large pad of paper.) This second branch of the solution appears to intersect the 45° line at A = B = e, although I recall that I was never able to confirm that to my complete satisfaction.

When the September issue of POPULAR COMPUTING arrived, I recalled this problem from the days of my plodding exuberance and, surprisingly, I located among my papers the curves I had drawn at that time.

I ... manage occasionally to find some time to work on problems in POPULAR COMPUTING...and I shall be delighted to continue contributing quarter-century-old solutions to "new" problems. Please consider this to be my first consecutive contribution thereto.

--ESL

Log 49	1.690196080028513661424432517185272386967144792647931
ln 49	3.891820298110626610210705486886359459274169459163722
$\sqrt[3]{49}$	3.659305710022971517238073310119408263487103668843324
$\sqrt[10]{49}$	1.475773161594552069276916695632244106544093613740204
$\sqrt[100]{49}$	1.039685437002590089820908637716288500930019285579133
49^{100}	10461838291314357175018899611816813659819188550170233
	65995014008403512576742426225177438261490936405029306
	52482525463141740631803436835911881507542673398165346
	37456120001
e^{49}	1907346572495099690525.099840953848447388189730543783
	402475234709367836008461905175
π^{49}	2292681686941669566432911.138212178849756827972223092
	078733698018488591824484598
$\tan^{-1}$ 49	1.550390996108358532788593351626825152761205226472767

How's Your Algebra?

A row of houses is numbered consecutively: 1, 2, 3, 4,... A resident of one of the houses notices that the sum of the numbers on one side of him is the same as on the other side. Thus, he could live in house numbered 6, with a total of 8 houses. What are the other possibilities?

A train goes from A to B from 1:00 to 5:00 P.M. Another train goes from B to A from 2:00 to 5:30 P.M. When do they pass each other?

Jules and Fred make a bicycle trip of 300 miles. Jules cycles east; Fred cycles west. They start their trips at the same time. When they pass each other, Jules takes 6 2/3 hours to complete his trip, and Fred takes 15 hours to complete his trip. What are their speeds?

One mile upstream from his starting point, a rower passed a log floating with the current. After rowing upstream for one more hour, he rowed back and reached his starting point just as the log arrived. How fast was the current flowing?

Find the price of eggs, if giving two less for 12¢ increases the cost of 100 eggs by 20¢

Solve this system:

$$\begin{cases} A + B + C = 1 \\ A^2 + B^2 + C^2 = 2 \\ A^3 + B^3 + C^3 = 1 \end{cases}$$

Sample problems from

The ALGEBRA Book

--to be published soon.

Solving time: one hour.

Problem Solution

Problem 156 (PC46-11) was the following:

For the system
$$\left.\begin{aligned} X + Y &= N \\ (X + \sqrt{X})(Y + \sqrt{Y}) &= 4N \end{aligned}\right\}$$

Find the value of N for which X is the smallest.

This was proposed as a problem in double bracketing. It was suggested that the system be solved for X for a given N by bracketing, and then a secondary bracketing be done on N to determine the minimum X. All of this would require extensive computation, which was the point of the problem.

However, as frequently happens, a good computer problem yields to better analysis. Herman P. Robinson demolished the "computer" problem as follows:

You have two equations and three unknowns, and the third "equation" is the response of the eyeball to a minimum value--not very good because the minimum is broad. I would solve as follows. Eliminate N, obtaining:

$$(X + \sqrt{X})(Y + \sqrt{Y}) = 4X + 4Y \qquad (1)$$

Replace X by $1/u^2$ and Y by $1/v^2$ and obtain

$$(u + 1)(v + 1) = 4(u^2 + v^2). \qquad (2)$$

Where we wanted the minimum value of X, we want the maximum value of u. Differentiate with respect to v:

$$(v + 1)u' + (u + 1) = 4(2uu' + 2v).$$

When u is a maximum, u' is 0, so

$$u + 1 = 8v$$

$$\text{or } v + 1 = (u + 9)/8 \qquad (3)$$

Combining (2) and (3) gives

$$(u + 1)(u + 9)/8 = 4(u^2 + (u + 1)^2/64$$

$$\text{or } 63u^2 - 18u - 17 = 0$$

$$\text{Then } u = (3 + 8\sqrt{2})/21$$

$$v = (u + 1)/8 = (3 + \sqrt{2})/21$$

$$X = 1/u^2 = (1233 - 432\sqrt{2})/289$$

$$Y = 1/v^2 = 99 - 54\sqrt{2}$$

$$N = X + Y = (29844 - 16038\sqrt{2})/289.$$

And this gives:

$X = 2.15245\ 58514\ 00771\ 41494\ 00362\ 32323\ 21948\ 11692$

$Y = 22.63246\ 76318\ 52867\ 36470\ 88088\ 92676\ 30375\ 72377$

$N = 24.78492\ 34832\ 53638\ 77964\ 88451\ 24999\ 52323\ 84069$

Mr. Robinson then goes on to offer two problems that may be more suitable for computation:

A

$$\left\{\begin{aligned} e^x + e^y + e^z &= 9 \\ x^2 + y^2 + z^2 &= 4 \\ y &= 0.25 \cosh x. \end{aligned}\right.$$

B

$$\left\{\begin{aligned} \sqrt{x} + \sqrt{y} - \sqrt{z} &= 1 \\ x + y + z &= 3 \\ x^2 + y^2 + z^2 &= 5. \end{aligned}\right.$$

Meanwhile, another problem has emerged that may lend itself to double bracketing. For the function:

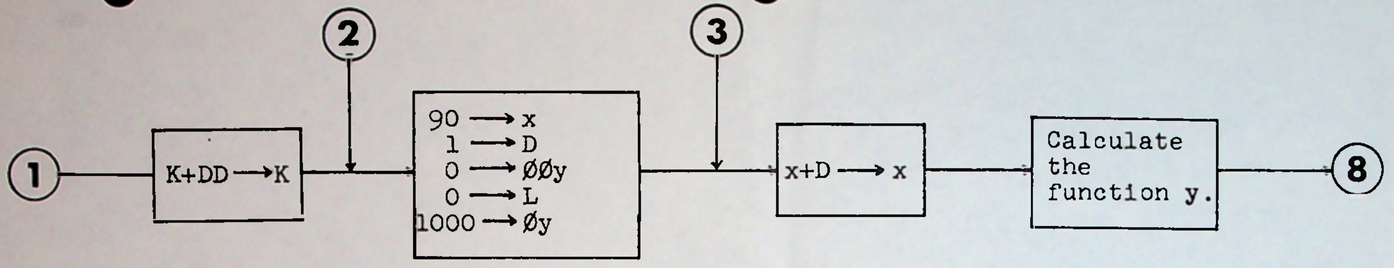
$$y = \ln(\sqrt{x} + K)^3 - \log(\sqrt{x} - K)^3$$

for a given value of K, there is an x that makes y a minimum. The problem is to find the value of K for which the minimum x is 100 (or any other specific constant). A possible flowchart is shown. For each value of K, the bracketing on x proceeds from Reference 2 to Reference 5; the bracketing on K proceeds from Reference 1 to the Halt. The value against which L is tested is the limit of precision on each x.

The original problem was suggested as material on which to use double bracketing; that is, to use the bracketing process to locate the X value for each N, and a superimposed bracketing process to find N. However, Richard Hamming, of the Naval Postgraduate School, has furnished an analytic solution:

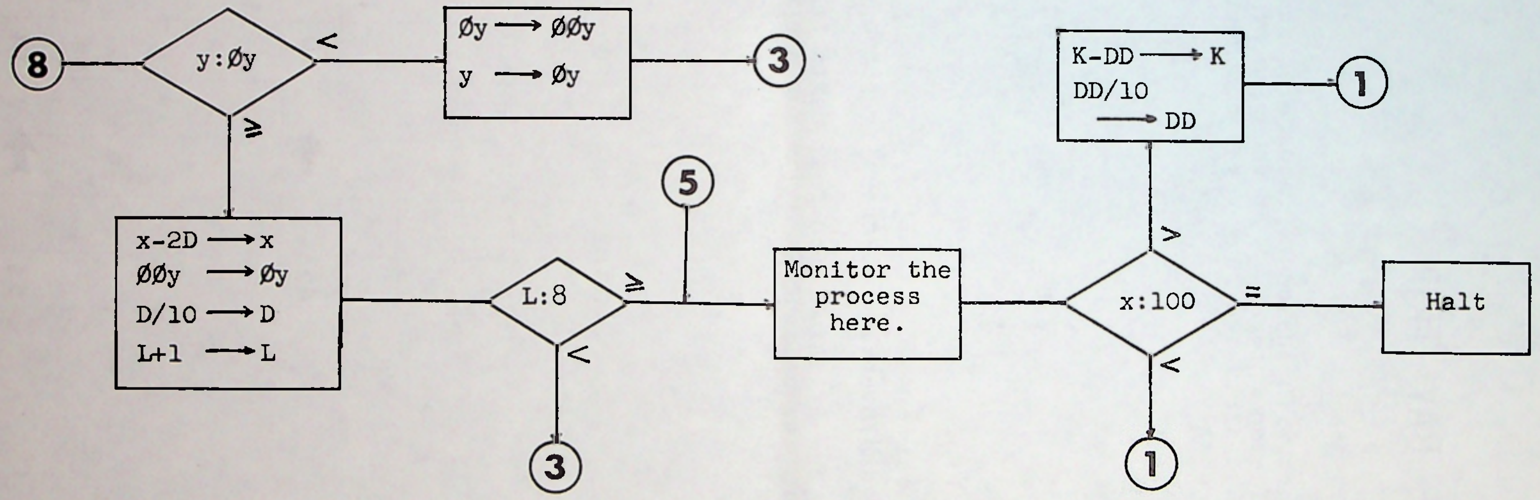
$$N = \frac{29844 - 16038\sqrt{2}}{289}$$

$$= 24.784923483253$$



DOUBLE BRACKETING on

$$y = \ln(\sqrt{x} + K)^3 - \log(\sqrt{x} - K)^3$$

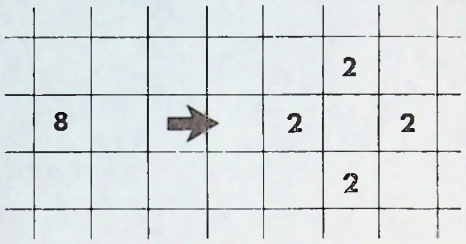


D is the increment while bracketing on x.
DD is the increment while bracketing on K.
Øy is "old y"
ØØy is the second oldest y.

Initialize K and DD (say, to 3.5 and .5) before entering at Reference 1.

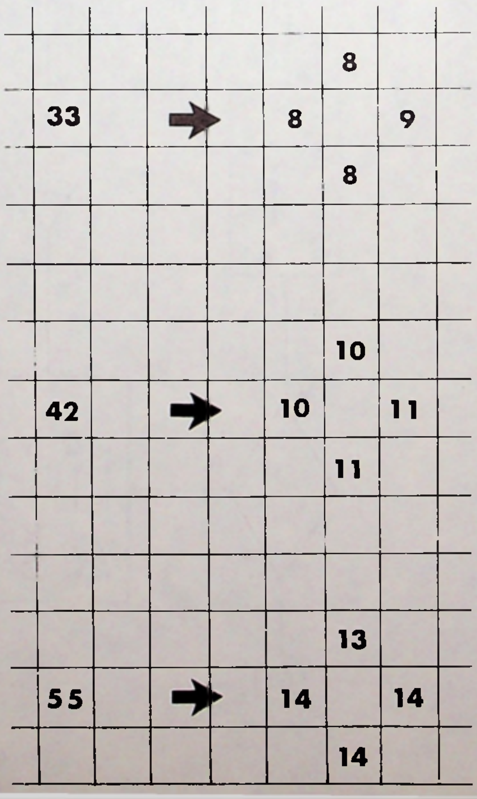
REPULSE BAY TRIP

Start with the center square of an array of squares; it contains an integer. At the time of a move, the number is distributed four ways, to the squares surrounding it orthogonally, thus:

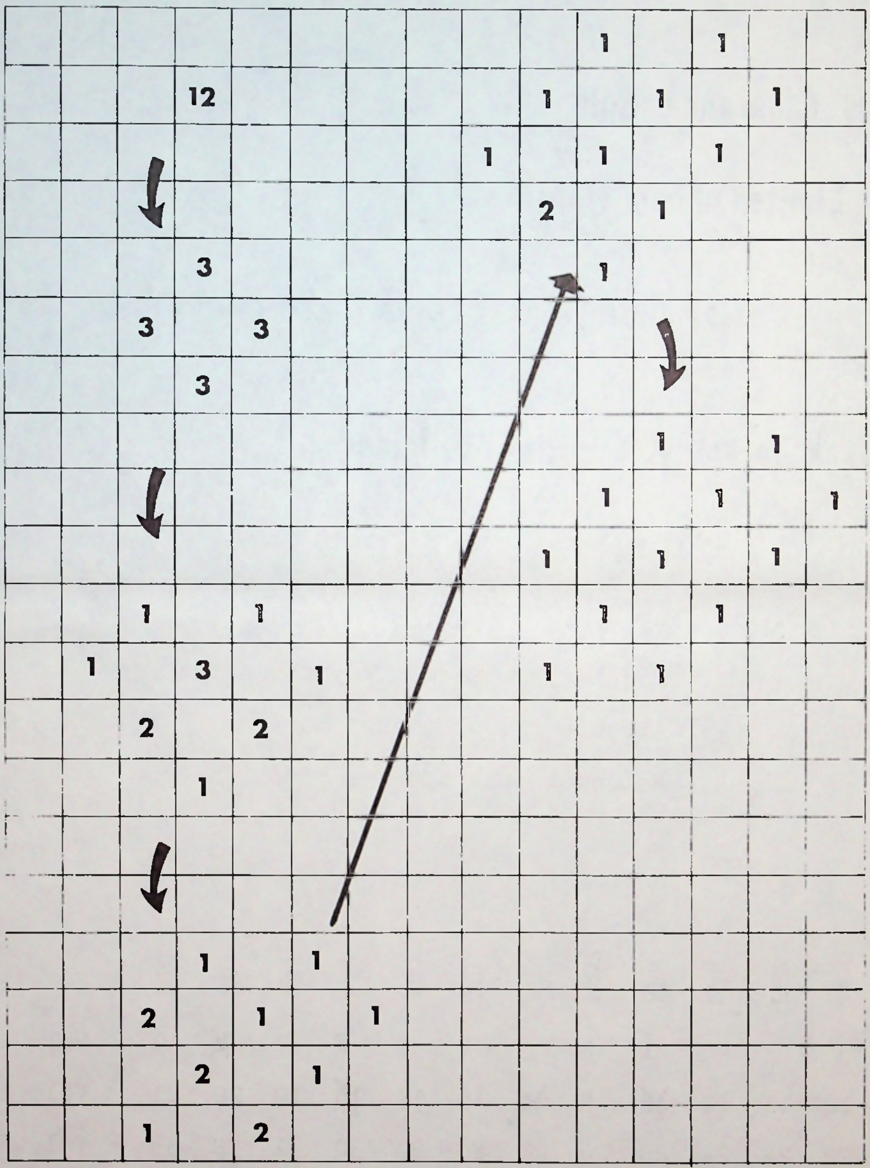


(sort of an inverse heat-transfer procedure.)

If the division into four parts has a non-zero remainder, the balance is distributed east, south, and west in that order, as in these examples:



The trip starts with one number in one square, and ends when the only numbers remaining are 1's. All squares containing numbers move at once. Thus, starting with 12, there will be five moves, as follows:



In general, the larger the starting number, the more moves the trip will take, and the larger the resulting pattern of 1's.

For small starting numbers, this procedure can be carried out by hand, but for larger numbers, this appears to be a computer problem.

Inventory Clearance Sale

Limited Time Only

Complete Your File of *Popular Computing*

Most Back Issues Are Still Available:

On orders received up to June 30, 1977:

1. For 10 or more of the issues that are still available.
2. $1.00 (U.S.) per copy.
3. Chosen from issues 1 through 39 only.
4. Mailed to one address by surface mail.

JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC		
			1	2	3	4	5	~~6~~	7	~~8~~	9	Vol. 1	1973
10	11	12	~~13~~	14	~~15~~	16	17	~~18~~	19	20	21	Vol. 2	1974
22	23	24	25	26	27	28	29	30	31	32	33	Vol. 3	1975
~~34~~	35	36	37	38	39	40	41	42	43	44	45	Vol. 4	1976
46	47	48	49	50	51	52	53	54	55	56	57	Vol. 5	1977

The Game of FIVES

FIVES (known also as Go-Moku and Go-Bang) is a game like Tic-Tac-Toe, but played on an infinite grid, with the objective of getting 5 in a row in any direction. Game courtesy calls for announcing "three" when an open string of three in a row is attained. Games between humans tend to be short (40 half moves would be quite long).

The game shown here is one played between a human (X) and a computer program (O), won by the program on the 20th half move. It is instructive to play off the game move by move.

			0	1	2	3	4	5	
1					X_{17}				
2			O_{20}	O_{14}	O_{16}	O_{18}	O_{4}		
3				X_{13}	O_{10}	O_{2}	X_{3}		
4				X_{9}	O_{6}	X_{5}	X_{1}		
5				X_{7}	O_{8}	O_{12}			
6				X_{19}	X_{11}		X_{15}		
7									

	X	O	
	4-4	3-3	
	3-4	2-4	
	4-3	4-2	"three"
	5-1	5-2	
	4-1	3-2	"three"
	6-2	5-3	
"three"	3-1	2-1	
	6-4	2-2	"three" (and a four)
	1-2	2-3	(now an open four)
	6-1	2-0	win

Considering the basic simplicity of the game, it would seem to be attractive to the writers of computer games programs. Yet the game shown is possibly the only recorded machine/human game in the literature.

The GE-225 had a Fives program in its library, but the program was limited to a 10 x 10 grid and was poorly written (for example, it would keep on moving after either player had won). A 10 x 10 grid is sufficient for most games, provided that the first player starts at one of the center squares, but it would not be difficult to allow for play that extends far from the center.

Fives is an open board game, with no random element and no hidden information. The following list possibly summarizes all that is known about it:

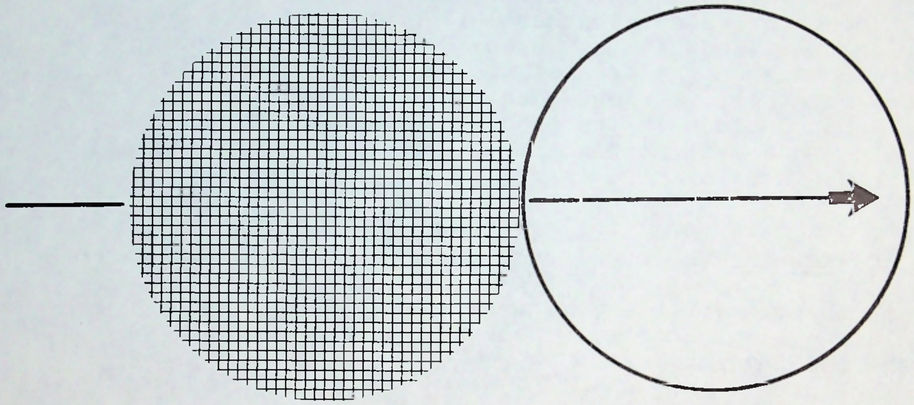
1. Between human players, some people are consistently better than others, which seems to imply that there is a winning strategy.

2. After the first move, moves must be made adjacent to previous moves; that is, one must play where the action is.

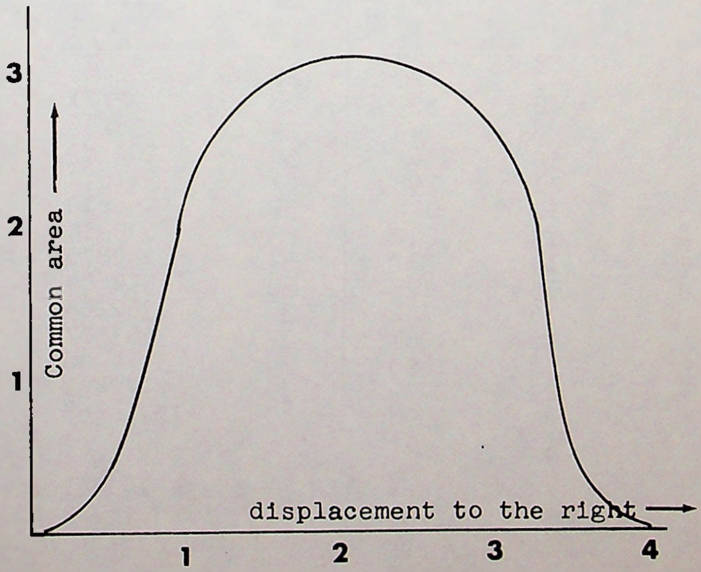
3. As far as possible, each move must be made to do double duty: to foster one's own goals and to thwart the opponent's goals.

As with other open board games, it is not necessary for the program to review the entire board situation after each play, but only those situations that have been altered by the opponent's move. This principle is particularly attractive in Fives, since moves don't react at a distance (as they do in chess, for example) but affect only nearby squares.

Eclipse



A circle of radius one is about to eclipse another similar circle. In terms of the horizontal displacement of the circle doing the eclipsing, what portion of the other circle is covered? That is, as the crosshatched circle moves to the right 4 units, what is the area that is common to both circles? Or, what is the shape of the curve:



This is a special case and elaboration of Mrs. Miniver's problem, which appeared in the Graham Dial magazine in February, 1945, submitted by Otto Gruenberger. In the book Mrs. Miniver there appears:

> "She saw every relationship as a pair of intersecting circles. It would seem at first glance that the more they overlapped the better the relationship; but this is not so. Beyond a certain point, the law of diminishing returns sets in, and there are not enough private resources left on either side to enrich the life that is shared. Probably perfection is reached when the area of the two outer crescents, added together, is exactly equal to that of the leaf-shaped piece in the middle. On paper there must be some neat mathematical formula for arriving at this; in life, none."

The problem is No. 3 in Ingenious Mathematical Problems and Methods (L. A. Graham, Dover Publications, 1959).

As a computing problem, we want a subroutine, whose argument is the displacement, which returns the common area to the calling routine.

The following sequence:

1	3	21	5396
2	4	22	8093
3	5	23	12139
4	7	24	18208
5	10	25	27311
6	14	26	40966
7	20	27	61448
8	29	28	92171
9	43	29	138256
10	64	30	207383
11	95	31	311074
12	142	32	466610
13	212	33	699914
14	317	34	1049870
15	475	35	1574804
16	712	36	2362205
17	1067	37	3543307
18	1600	38	5314960
19	2399	39	7972439
20	3598	40	11958658

is generated by a simple algorithm. (1) Find that algorithm. (2) Find the 1000th term of the sequence. (3) Estimate what fraction of the terms are perfect squares.